

FIG. 1

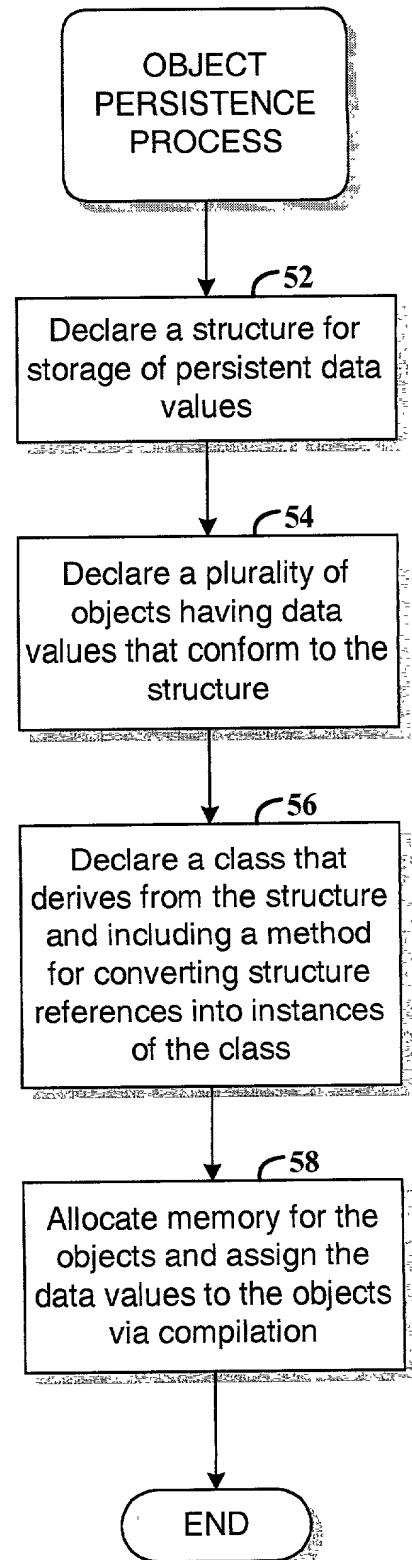


FIG. 2

102

*example
structure*

```
struct SPerson
{
    char *m_name;
    unsigned long m_dob;
};
```

104

*example
class*

```
class CPerson : private SPerson
{
public:
    static CPerson &convertTo( SPerson &from )
    {
        return static_cast< CPerson & >( from );
    }
};
```

FIG. 3

FIG. 4

106

*example
code*

```
{
    ...
    SPerson myData =
    {
        "John Smith",
        19620612
    }
    CPerson &me = CPerson::convertTo( myData );
    ...
}
```

FIG. 5

```

class CPersonTraits
{
public:
    typedef CPerson type;
    typedef SPerson data;
    typedef SPerson base;
};

class CCarTraits
{
public:
    typedef CCar type;
    typedef SCar data;
    typedef SCar base;
};

struct SCar
{
    char *m_make;
    char *m_model;
    CPersonTraits::data *m_owner;
};

class CPerson : private CPersonTraits::base
{
public:
    typedef CPersonTraits traits;

    static traits::type &convertTo( traits::data &from )
    {
        return static_cast<traits::type &>( from );
    }
};

class CCar : private CCarTraits::base
{
public:
    typedef CCarTraits traits;

    static traits::type &convertTo( traits::data &from )
    {
        return static_cast<traits::type &>( from );
    }

    CPerson &owner()
    {
        return CPerson::convertTo( *m_owner );
    }
};

```

*example
traits*

108

FIG. 6

110

*example
collection*

```
template< class CollectionTraits >
struct SStructCollection
{
    unsigned int m_size;
    CollectionTraits::collectionData *m_first;
};
```

FIG. 7

112

```
template< class TypeTraits, class Initializer = CNullStructCollectionInitializer<
TypeTraits > >
class CStructCollectionTraits
{
public:
    typedef TypeTraits::data collectionData;
    typedef TypeTraits::type collectionType;

    typedef CStructCollection< CStructCollectionTraits< TypeTraits, Initializer >
        > type;
    typedef SStructCollection< CStructCollectionTraits< TypeTraits, Initializer >
        > base;
    typedef SStructCollection< CStructCollectionTraits< TypeTraits, Initializer >
        > data;

    typedef CCollectionIterator< TypeTraits > iterator;

    typedef Initializer initializer;
};
```

FIG. 8

*example
collection
iterator*

```

template< class TypeTraits >
class CCollectionIterator
{
public:
    typedef TypeTraits::data data;
    typedef TypeTraits::type type;

    CCollectionIterator( data *begin ) : m_p( begin ) {}
    CCollectionIterator( const CCollectionIterator< TypeTraits > &from ) : m_p(
        from.m_p ) {}
    CCollectionIterator &operator =( const CCollectionIterator< TypeTraits >
        &from )
    {
        m_p = from.m_p;

        return *this;
    }
    ~CCollectionIterator() {}

    type *operator ->()
    {
        return &type::convertTo( *m_p );
    }

    type &operator *()
    {
        return type::convertTo( *m_p );
    }

    CCollectionIterator< TypeTraits > &operator ++()
    {
        ++m_p;

        return *this;
    }
}

```

FIG. 9A

*example
collection
iterator*

```
data *pointer()
{
    return m_p;
}

CCollectionIterator< TypeTraits > operator +( int addend ) const
{
    return m_p + addend;
}

CCollectionIterator< TypeTraits > operator -( int addend ) const
{
    return m_p - addend;
}

bool operator !=( const CCollectionIterator< TypeTraits > &to ) const
{
    return m_p != to.m_p;
}

bool operator <( const CCollectionIterator< TypeTraits > &to ) const
{
    return m_p < to.m_p;
}

private:
    data *m_p;
};
```

FIG. 9B

```

template< class TypeTraits >
class CStructCollectionInitializer
{
public:
    typedef TypeTraits::collection collection;

    inline CStructCollectionInitializer( collection::traits::data &collectionData )
    {
        m_collection( collection::convertTo( collectionData ) )
    }

    for ( collection::traits::iterator i = m_collection.begin(); i < m_collection.end();
          ++i )
    {
        new( i.pointer() ) TypeTraits::type;
    }
}

inline ~CStructCollectionInitializer()
{
    for ( collection::traits::iterator i = m_collection.begin(); i < m_collection.end();
          ++i )
    {
        typedef collection::traits::collectionType collectionType;

        i->collectionType::~collectionType();
    }
}

private:
    collection &m_collection;
};

```

*example
collection
initializer*

FIG. 10

*example
pre-processor
macro*

```
#define COLLECTION_BEGIN( TypeTraits, CollectionName ) \
TypeTraits::data CollectionName##Collection[] = \
{ \
    #define COLLECTION_END( TypeTraits, CollectionName ) \
}; \
TypeTraits::collection::traits::data CollectionName = \
{ \
    sizeof( CollectionName##Collection ) / sizeof( TypeTraits::data ), \
    CollectionName##Collection \
}; \
TypeTraits::collection::traits::initializer CollectionName##Initializer( \
    CollectionName );
```

FIG. 11

*example
pre-processor
forward referencing
macro*

```
#define COLLECTION_FORWARD( TypeTraits, CollectionName ) \
TypeTraits::data CollectionName##Collection[];
```

FIG. 12

122

*example
pre-processor
inter-collection
reference macro*

```
#define ENTRY_REF( CollectionName, Element ) \  
&CollectionName##Collection[ ( Element ) ]
```

FIG. 13

124

*example
pre-processor
empty collection
macro*

```
#define COLLECTION_NO_ENTRIES( TypeTraits, CollectionName ) \  
TypeTraits::collection::traits::data CollectionName = { 0, 0 };
```

FIG. 14

125

*example aggregate
supporting
virtual data members*

```
template< class TypeTraits, unsigned int N = 1 >  
struct SVirtualSupport  
{  
    struct  
    {  
        void *m_vptrs[ N ];  
        TypeTraits::base m_data;  
    };  
    void ( *m_initializer )( void *where );  
};
```

FIG. 15A

126

```
template< class TypeTraits >
class CInstanceInitializer
{
public:
    static void initialize( void *where )
    {
        new( where ) TypeTraits::type;
    }
};
```

*example
vfpointer
initialization*

FIG. 15B

128

```
class CCarTraits
{
public:
    typedef CCar type;
    typedef SVirtualSupport< CCarTraits > data;
    typedef SCar base;
    typedef CInstanceInitializer< CCarTraits > initializer;
};
```

*example
vfpointer
initialization*

FIG. 15C

130

```
{
...
CCar::traits::data carData =
{
    { { 0 }, { "Ford", "Escort", &personCollection[ 0 ] }, CCarTraits::initializer }
};

( *carData.m_initializer )( carData ); // initialize car instance.

CCar &car = CCar::convertTo( carData ); // produce a real CCar instance
                                         // from the already initialized data.

...
}
```

*example
vfpointer
initialization*

FIG. 15D